vLab: an Eclipse-based Virtual Laboratory

A. Riccioni, E. Denti – Dip. Elettronica, Informatica e Sistemistica, Università di Bologna, Italy.

{anna.riccioni, enrico.denti}@unibo.it

Abstract

In modern education, and especially in the Computer Engineering field, integrating conceptual understanding with technical skills assumes a strategic importance. The new Information and Communication Technologies can help in addressing this issue allowing the implementation and the delivery of new advanced educational tools, such as virtual labs, for fostering teaching and learning.
In this work, we first outline the requirements a virtual lab must meet to effectively support students through the whole learning process typical of the Computer Engineering education. After validating the chosen approach by experimenting with a first, simplified prototype, tailored to the case study of the Information Security Technologies course, we refine our guidelines, then design a suitable architecture for the final version of virtual lab and finally propose an implementation based on Eclipse technology.

1. Introduction

During the last years the interest on distance learning techniques has been growing steadily, and computer-based training is quickly becoming more and more accepted.

The adoption of new technologies can improve traditional education, increasing its personalization capabilities thus making it more effective. In fact, ICTs can help in supporting different learning styles and phases: according to various learning theories, this is essential for a successful training [1-4].

It is widely acknowledged [5] that, in order to be effective, a learning experience must include two main steps: the first, known as descending curriculum, is centered on notion transmission from the instructor to students, while the second is focused on an ascending curriculum, according to a constructivist approach. There, learners are called to play a more active role, building their own knowledge by practicing on case studies, verifying and applying concepts to new scopes, designing and evaluating new scenarios [6]. The transposition of conceptual knowledge into practice helps students to reinforce learning, gain greater insight into subjects and acquire practical skills and long-lasting proficiency [7-13].

Such two key steps can be effectively fostered through a blended learning approach, intended as a combination of traditional synchronous face-to-face lectures and asynchronous, autonomous work sessions where students practice on a proper educational tool.

In this paper we refer to the context of Computer Engineering education, where integrating conceptual understanding with technical skills assumes a strategic importance. In this perspective, the goal of our research is to design a virtual lab for supporting students through their learning experiences.

This paper is organized as follows. In the first part, we first define the general requirements a virtual lab must meet to achieve its goal (Section 2), and briefly report on the results of our early experiments in vLab 1.0 (Section 2.1). Then, from the try-out results, we refine our project guidelines for the design of the final version of virtual lab (Section 2.2): accordingly, we discuss a suitable architecture (Section 3), and propose

an implementation based on Eclipse technologies (Section 3.1) with the related roadmap (Section 3.2). Conclusions and related work are reported in Section 4.

2. Virtual labs

In Computer Engineering education, where the mastery of technical content is a necessary but not sufficient requisite, the need for a virtual lab for fostering students practical abilities is particularly evident. In fact, they need both strong skills in problem-solving, design, judgment and decision-making, and proficiency in programming languages to be able to design and implement a well dimensioned software system.

In a problem-based learning process, once that the requirements and the domain have been settled, the student follows an incremental process cycling through four main stages:

- analyze the problem
- sketch a solution
- verify how the model fits the requirements
- design, implement and test a prototype.

Students can accomplish the first two steps by outlining a simplified but representative version of their proposed solution [8, 14]. To this end, a virtual lab must expose CAD-like functionalities, providing users with an interactive editor and a context-specific palette exposing the available model components. Such a tool allows the students to define the structure of the considered system.

Furthermore, the virtual lab must include an engine for the interpretation and repeated execution of the system model, so that learners can perform simulations to investigate the system behavior and assess the suitability of the envisioned solution. Observing the simulation results makes it easier to identify critical aspects and predict the response to stimuli in a given working condition. The interpretation of the simulated results also helps students to evaluate their model, estimate the needed resources and adjust the system sizing [1, 9]. Thus, a virtual lab must also include a data gathering service and several data analysis tools.

Once the model has been checked and found appropriate, users must be able to build a software prototype of the designed system. The virtual lab must thus support learners in this step, for instance by yielding extracts of sample code. A close link between the virtual lab and an Integrated Development Environment would also be an added value, enabling students to autonomously modify, extend and reuse the partial code automatically generated during the exercitation in different contexts.

Finally, a tracking service is needed in order to provide users with prompt feedback related to their activities; in fact, a real-time advising helps students in quickly identifying errors and fixing them, thus simplifies and speeds up the learning process [15]. Wizards, online help and simplified access to external additional resources, such as technical documentation or reference standards related to the problem domain, are further benefits for learners.

Our purpose is to design and implement a virtual lab supporting all the above features; in particular in the context of the Information Security Technologies course hold at the University of Bologna.

A first, simplified version of virtual lab, vLab 1.0, was developed in 2005, aimed at supporting rapid prototyping; however, that early version intentionally met only a subset of the whole requirements. Thus, we complete our guidelines according to the results obtained through vLab 1.0 experimentation, and consequently refine vLab 2.0 analysis and design.

2.1 vLab 1.0 experimentation

vLab 1.0 graphical user interface, shown in Fig.1, is composed by three main areas: the larger one shows a schematic representation of the current exercise in terms of interconnected blocks, each representing a security mechanism, and allows students to set the system parameters and run experiments. Every block is associated to a (set of) function, and can have inputs, outputs and configuring parameters. The second area is devoted to feedback about the current experiment: for each component in the system model, the window reports notes about data dimension and execution time in case of success or error messages in case of failure. Finally, the third view displays an educational sample Java code of the ongoing exercise.

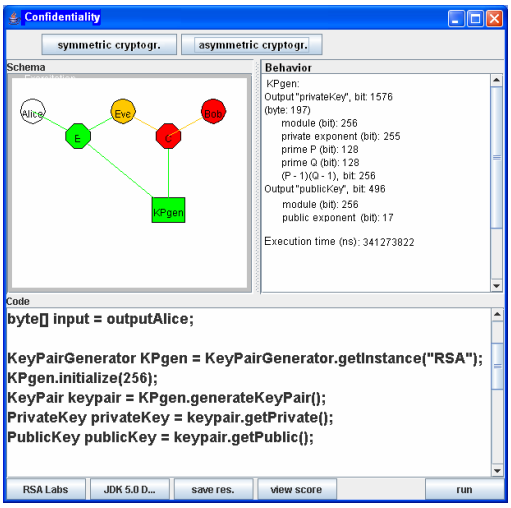

Fig. 1 - vLab 1.0 screenshot.

vLab 1.0 was designed as a first, simplified version of the whole virtual lab project. Its main goal was to leverage the upcoming edition of the Information Security Technologies course as a preliminary acceptance test and as a validation of the chosen approach. For this reason, along with strict time constraints, vLab 1.0 was most tailored to rapid prototyping: it is based on a simple three-tier architecture and adopts Java 5.0 as reference technology.

The two-year experimentation conducted by making vLab 1.0 available to around 200 students confirmed the effectiveness of the chosen approach and served as a successful acceptance test. This try-out also remarked the urgent need for a more extensive support to students' active role: learners should be allowed to build and modify the system models, to extend the component library and to contribute their own data analysis tools to the environment. Usability and expressivity of the lab should also be improved adding supplementary features such as the ability to save and restore working sessions, to access a comprehensive online help and to switch between different detail levels within the graphical schema.

2.2 vLab 2.0 guidelines

As outlined in the introduction, vLab 2.0 is inspired to the constructivist approach, which highlights the strategic importance of providing learners with different tools to let them autonomously pursue their formative goals, following their own interests and logics [9].

In addition, the virtual lab must be open to upgrades as well as to users' contributes, so that students can be actors in their own learning process by freely modifying and expanding the educational tools according to their changing needs. To this end, they should be allowed to yield and share elaborates, building new exercitation schemas, and integrate within the virtual lab their own data analysis tools to fit their own specific needs.

In line with these goals, vLab 2.0 supplies users with a wide variety of facilities shown in Fig.2 [16].

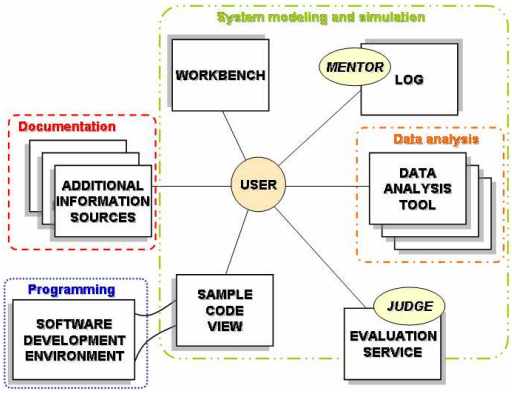

Fig.2 - vLab 2.0 as a set of features and automatic actors.

The different features can be grouped in default sets tailored to specific objectives, like system modeling and simulation, data analysis, documentation and programming environment, each representing a working mode. vLab 2.0 GUI should then include the proper tool collection based on the chosen working mode; users should be allowed to switch between the several chances and modify the set configuration varying the included tools.

Two special functionalities in Figure 2 have a particular relevance. *Mentor* is an automatic learner assistant that during modeling and simulation steps observes the users activities and consequently provides feedback notifying the students with general info, warning and error messages; sometimes it can also suggest how to solve an identified known problem. This feature represents the evolution of vLab 1.0 prompt feedback service: the main difference is the separation between general advices and experiment-

related information. While vLab 1.0 presents both kind of feedback through the same view, vLab 2.0 redirects the gathered data to suitable data analysis tools.

The *judge* is the second automatic assistant, aimed at guiding the users through the virtual lab. Its main responsibility is the evaluation of the users' learning processes: knowing the learners' starting competence level, the judge estimates their performances and suggests the task or activity which is most suited to their formative goal. In the future, such a feature could be tailored to evaluate also users cooperation ampliating its scope outside the virtual lab to include external support services.

3 vLab 2.0 design model

In order to address the above requirements, and in particular the two major issues –extensibility and integration with several tools- an effective choice is to adopt a plug-in based architecture. Accordingly, the tools to be integrated in the virtual lab will be built as plug-ins or sets of plug-in implementing a given interface: possibly, they could publish their own in order to be extensible too, enabling vLab to be easily upgraded or enhanced with new features.

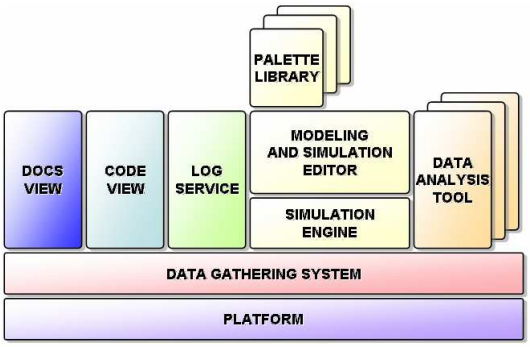

Fig.3 - vLab 2.0 plug-in architecture.

In addition, we assume that the system is built on top of a platform providing some basic coordination and communication services, as these are essential for collecting and managing info from the different components. The resulting architecture is depicted in Fig.3.

Each feature within the virtual lab is achieved through a dedicated plug-in or set of plug-ins, depending on its complexity. For instance, the modeling editor and the simulator engine could be designed as a combination of two interacting plug-ins, in order to uncouple the graphics management from the execution of a configured experiment.

Furthermore, in order to specialize the virtual lab to different formative domains, it is crucial to be able to configure the model editor with several context-specific component palettes. The various palette libraries can thus correspond to distinct plug-ins, and be dynamically added to the virtual lab.

A similar idea also applies to the data analysis tools since users must be able to design, integrate and share their custom components. Further plug-ins provide the logging service needed by the mentor and the sample code view.

In order to consistently cooperate and synchronize, all such plug-ins need to share a common reference model. vLab 2.0 reference abstractions are the Finite State Machine and the Block Diagram: each system can be represented in terms of interconnected blocks, and each block, during schema construction, configuration and execution, goes through several states. Furthermore, the block diagram model allows the exploration of the system structure at different detail levels through a zooming mechanism. This is a really appreciated facility within a simulation tool: it helps learners to deeply analyze the system behavior, providing a more effective, customizable and comprehensive didactic experience.

3.1 vLab 2.0 and Eclipse

Given the above plug-in based reference architecture, Eclipse platform was a natural candidate for building the virtual lab. In fact, it is open source, based on plug-ins, and adopted by many advanced projects. Moreover, some third-party plug-ins offer a comprehensive support for software design, development and deploy, and thus represent a valuable contribute to simplify and speed up our implementation. As a further benefit, Eclipse is a well-known reality in both academic and professional scope, and often computer engineers start learning how to deal with it during their first years at University. For this reason students are supposed to be quite familiar with vLab 2.0 user interface, Eclipse

IDE and its standard features.

Fig.4 outlines a specialization of vLab 2.0 plug-in architecture to Eclipse technology.

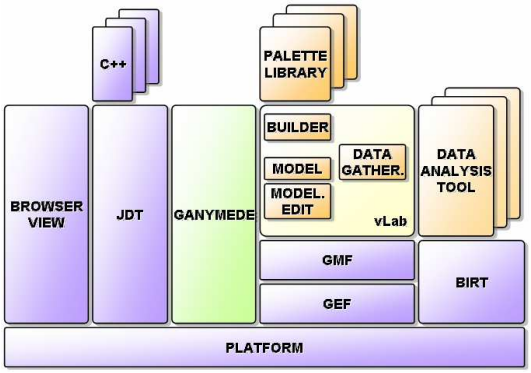

Fig.4 - vLab 2.0 and Eclipse.

The system is rooted on the Eclipse platform, which supplies a comprehensive set of features, including some coordination, communication and data management mechanisms that in vLab 1.0 were spread over several layers.

The core of the virtual lab can be implemented as an *Eclipse feature*, intended as a structure that bundles together a set of plug-ins concurring in the development of a complex functionality. The key component within the feature will be the modeling and simulation editor: in order to simplify and expedite its implementation we can rely on two Eclipse projects, the Graphical Modeling Framework (GMF) and the Eclipse Modeling Framework (EMF). In fact, EMF allows to design the finite state machine reference model, and then, through GMF, we can automatically obtain a basic editor focused on this abstraction.

The generated editor relies on the Graphical Editing Framework (GEF), an thus inherits a set of facilities which greatly simplify the graphical management. Furthermore, the adoption of GMF makes it easier to extend the basic editor features; as the generated model is organized in two separated modules: `vlab.model` represents the reference data model and application logic, while `vlab.model.edit` holds the control logic. In our prototype, we include in this module also the GEF edit parts needed to describe the model behavior within the editor and also the edit policies that statically define the simulator engine's working procedures. As a result, it is possible to validate the users' actions aimed at build and modify a schema within the editor referring to the edit parts listed in the `vlab.model` plug-in. On the other hand, the consequences related to the execution of a component of the schema are ruled by the edit policies included within the same plug-in.

Eclipse platform provides a built-in support, based on the Property Descriptor mechanism, for communication and synchronization between the application logic elements. Leaning on such a facility grants a robust and immediate communication and notification infrastructure that avoid the complexity of the manual management of a message exchange protocol or observable-observer pattern implementation. Within vLab 2.0, the Property Descriptor facility is employed in two steps: the first is as a setting interface, through the dedicated Eclipse Property view, that students fill when arranging their experiment. Then, during the simulation phase, the Property Descriptor assumes the responsibility of propagating the proper notification to the interested listener, in order to maintain a complete and consistent model.

BIRT, a comprehensive Eclipse-based open source reporting system for web applications, is a significant example of a ready-to-use software component that can simplify the implementation of several vLab features. In fact BIRT, in its plug-in form, can be included in the virtual lab in order to provide an extensive support for creating advanced data analysis tools. The BIRT project already includes facilities for users to autonomously build their reports, following the composition of a template by selecting primitive components from a palette, that is the same procedure that learners must follow when building their system models. It is also possible to extend the default palette with user-provided new elements: the same facility will be provided within the modeling editor, where users can extend the basic palette including their own schemas.

Another key feature in vLab 2.0 is the mentor, that can be implemented through a standard logger configured to receive notifications from the running instances of the reference domain model. An interesting tool tailored to log creation is Log4J, whose plug-

in version, called Ganymede, can easily be integrated within the Eclipse platform, and, as a result, within any Eclipse-based application or Rich Client Platform. In order to perform this integration, it is important to properly tie the exception handling within the vLab feature to the Ganymede logger listener: this task can be accomplished relying on the Eclipse platform socket communication support.

Finally, the Eclipse platform makes it possible to easily provide learners with the access to an advanced IDE where they can expand, modify, test and reuse the sample code generated within the virtual lab. Thanks to several plug-ins like JDT or C++DT, the Eclipse IDE can be configured in order to be used with different programming languages.

3.2 Roadmap

Our aim is to complete a first stable beta version of vLab 2.0 by January 2008, in order to be able to use it within the next edition of the Information Security Technologies course hold at the Engineering Faculty of the University of Bologna.

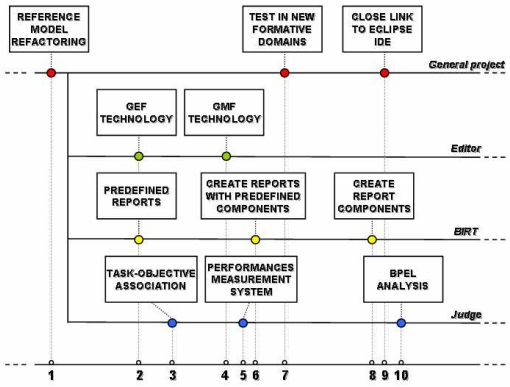

Fig.5 – Roadmap outline.

The first milestone, representing a complete refactoring of the reference model, is the preliminary step needed for achieve extensibility and, as a result, steadiness. Moving from this prior stage, the project development can be split in several guidelines proceeding in parallel.

The first is associated with the ongoing project expansion and application to other formative domains. In this context, the implementation of several context-specific palettes dedicated to different case studies is an interesting step forward, since it will make it possible to test the virtual lab effectiveness and . Relevant application could be related to logic design or network protocols.

The strong integration within vLab of the Eclipse Java IDE represents a further significant progress in order to fully support learners' programming activities. Furthermore, is an additional test for the virtual lab extensibility.

The editor development, due to its complexity, will proceed through two main steps. The first is aimed at investigating GEF fully potential for managing advanced graphical features, relying on a manually built model based on the block diagram abstraction. Then, we will switch to the GMF technology in order to gain a more maintainable standard representation of the reference model.

One further development direction is represented by BIRT integration within the virtual lab. As a first step, we plan to provide students with a few ready-to-use reports, composed by the instructors, among which they can choose. We will then extend the constructivist approach to the reporting activities, first letting students access the BIRT report composition perspective, so as to be able to create their own reports; then, depending on vLab and BIRT extensibility, enabling users to implement their own report component, sharing and integrating them within the report's palette.

The BPEL project hold within the Eclipse Community is another promising direction for realizing the judge intelligent tutoring system. Thus, another interesting guideline to follow could be related to investigating BPEL project advancements and to making plans for integrating it into vLab.

Finally, while analyzing a suitable approach in order to implement the intended judge, it is possible to plan and arrange the basic support structure needed for providing customization and evaluation. In fact, in order to measure users' competences starting level and their performances, it is important to defining a proper measurement system. In more detail, we will define a set of formative goals, relying on Bloom's taxonomy [17], and associate them to the appropriate tasks within the virtual lab. In this perspective, three key

issues arise: the first is the definition of a suitable measurement parameter set. Then, we will specify how to (quantitatively) measure the users' performances in terms of the established parameters. Lastly, we will formalize a measure integration aimed at providing a comprehensive evaluation of the learner's performances, depending on his starting level and on the formative goals associated to the considered activities.

As first step towards this goal, we intend to provide a significant link between formative goals and exercitations in order to allow a manual evaluation of the students' performances in the upcoming edition of the Information Security Technologies course.

4 Conclusions and related work

In this paper we discussed our approach to the design of a virtual lab for fostering teaching and learning. Our purpose is to define the requirements and the reference model such an educational tool must comply with in order to be suitable for use in different formative domains related to the Computer Engineering education. While testing our prototype, we will refer to the course on Information Security Technologies as a case study.

Several works have been carried out on similar goals: many authors have outlined and designed modeling frameworks or simulators suitable for education at various levels [1, 7, 8, 18, 19]. Furthermore, in order to achieve technical proficiency and to improve practical programming skills, students often benefit from the employ of advanced IDEs: thus, research efforts are also aimed at designing pedagogical support tools to be added to commonly used software development frameworks [14, 15, 18, 20]. These two positions are focused on supporting students only in one or two phases of their learning process – respectively modeling and simulation and prototype development-, disattending the constructivist approach. Thus, our purpose is to integrate these main directions in one comprehensive educational tool.

According to constructivism, our intended virtual lab has been designed to manage extensibility and provision of several tools, and to be strictly connected to an Integrated Development Environment. For these reasons, the Eclipse platform seemed a suitable implementing framework. The initially steep learning curve to acquire proficiency in the Eclipse technology is widely rewarded by the huge variety of advanced projects and third-parties plug-ins developed within the community: some of them, tailored to general needs such as reporting or logging facilities, can be integrated into the virtual lab in order to contribute to the required features. Others, oriented at assisting developers in building, deploying and managing software, can represent an useful support throughout the virtual lab implementation step.

vLab 2.0 is still an ongoing project. With respect to the roadmap in Fig.5, we currently have fully accomplished the first milestone, dedicated to the reference model refactoring, and we are still working on the second point, both for integrating BIRT predefined reports and for improving the GEF-based editor. The third and forth milestones represent our very next steps towards the foundation of the judge system and the editor refinement.

Our future efforts will be addressed both at improving vLab adding new advanced features and at dealing with new open and challenging issues. A key point is represented by users' activities evaluation, and is strictly connected to the presentation of customized feedback and the suggestion of new tasks based on users' profiles, formative objectives and activity history. This research direction will also imply an integration with the standard SCORM [21].

Bibliography

1. BÜCHNER P., NEHRIR M H., *A Block-Oriented PC-Based Simulation Tool For Teaching and Research in Electric Drives and Power Systems*, IEEE Transactions on Power Systems 6 (3): 1299-1304 (1991).
2. BROWN S. A., LAHOUD H. A., *An Examination of Innovative Online Lab Technologies*, SIGITE '05: Proceedings of the 6th conference on Information technology education: 65-70 (2005).
3. YANG T. A., YUE K. B., LIAW M., COLLINS G., *Design of a Distributed Computer Security Lab*, Journal of Computing Sciences in Colleges 20 (1): 332-346 (2004).
4. RIGBY S., DARK M., *Designing a Flexible, Multipurpose Remote Lab for the IT Curriculum*, SIGITE '06: Proceedings of the 7th conference on Information technology education: 161-164 (2006).
5. GUERRA L., *I Limiti della Qualità Didattica*, QueL: seminario di lancio dell'Iniziativa Nazionale per la Qualità nell'eLearning (in Italian, 2005).
6. GUERRA L., *L'elaborazione Didattica di Learning Objects*, Ricerche di Pedagogia e Didattica. Retrieved on June 2007 from http://rpd.cib.unibo.it/archive/00000015/ (in Italian, 2006).
7. CHUNG G. K., HARMON T. C., BAKER E. L., *The Impact of a Simulation-Based Learning Design Project on Student Learning*, IEEE Transactions on Education 44 (4): 390-398 (2001).
8. ALLWOOD J. M., COX B. M., LATIF S. S., *The Structured Development of Simulation-Based Learning Tools With an Example for the Taguchi Method*, IEEE Transactions on Education 44 (4): 347-353 (2001).
9. BLACK J. B., MCCLINTOCK R. O., *An Interpretation Construction Approach to Constructivist Design*, in B. Wilson (Ed.) Constructivist Learning Environments (1995).
10. GUIDORZI R., *e-Learning Projects at Bologna University: an Overview*, Proceedings of the International Conference on Networked e-Learning for European Universities (2003).
11. GUIDORZI R., DIVERSI R., COLIN M., LODOLI G., *A Constructivist Approach in Designign an e-Learning System Identification Course*, Preprints of the 7th IFAC Symposium on Advances in Control Education (2006).
12. KOLB D. A., *Experiential Learning experience as a source of learning and development*. New Jersey: Prentice Hall (1984).
13. KOLB D. A., BOYATZIS R. E., MAINEMELIS C., *Experiential Learning Theory: Previous Research and New Directions*, in R. J. Sternberg and L. F. Zhang (Eds.), Perspectives on Cognitive, Learning, and Thinking style (1999).
14. MA J., NICKERSON J. V., *Hands-on, Simulated, and Remote Laboratories: A Comparative Literature Review*, ACM Computing Surveys 38 (3): art. n. 7. DOI= http://doi.acm.org/10.1145/1132960.1132961 (2006).
15. DE BARROS L. N., DOS SANTOS MOTA A. P., DELGADO K. V., MATSUMOTO P. M., *A Tool for Programming Learning with Pedagogical Patterns*. In Proceedings of the 2005 OOPSLA Workshop on Eclipse Technology Exchange: 125-129 (2005).
16. LASCHI R., RICCIONI A., SUZZI P., *Learning by Doing: vLab, a Virtual Laboratory for Computer Engineering Education*. DET '07: Proceedings of the International Workshop on Distance Education Technologies: 127-132 (2007).
17. BLOOM B. S., *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain*. New York: David McKay Co Inc (1956).
18. LÉDECZI Á., BAKAY Á., MARÓTI M., VÖLGYESI P., NORDSTROM G., SPRINKLE J., KARSAI G., *Composing Domain-Specific Design Environments*. Computer 34 (11): 44-51 (2001).
19. BUCHER H. F., SCHULTZ A. J., KOFKE D. A., *An Eclipse-based Environment for Molecular Simulation*. In Proceedings of the 2005 OOPSLA Workshop on Eclipse Technology Exchange: 130-134 (2005).
20. REIS C., CARTWRIGHT R., *A Friendly Face for Eclipse*. In Proceedings of the 2003 OOPSLA Workshop on Eclipse Technology Exchange: 25-29 (2003).
21. ADVANCED DISTRIBUTED LEARNING, *SCORM 2004 3rd Edition*. Available from http://www.adl.gov (2006).